Personalized Medication and Activity Planning in PDDL+

Fares K. Alaboud and **Andrew Coles**
Department of Informatics, King's College London, UK
email: firstname.lastname@kcl.ac.uk

Abstract

The emergence of planners capable of reasoning with continuous dynamics, as expressed in PDDL+, has increased the range of problems that fall within the capabilities of PDDL planners. One such problem is planning patients' activities and medication regimes, considering non-linear medication pharmacokinetics. In this paper we explore the application of contemporary PDDL+ planners to this problem. To address their performance limitations, we present a linearize–validate cycle; tasks are solved by iterative refinement of a linear approximation of the domain, solved by a linear planner, then validated at each stage against the full non-linear semantics. In doing this we allow this domain to fall within the capabilities of current planners; and in our evaluation we use OPTIC to demonstrate this.

1 Introduction

One of the largest problems in healthcare is the incorrect consumption of medication. It is estimated that half of patients that are prescribed medication for chronic conditions do not consume their medication correctly (The Academy of Medical Sciences 2014). Most medication is prescribed in a way that expects the patient to follow a standard routine. This is done in order to help the patient stay compliant, and at the same time to consume the medication in a way that does not endanger the patient – often, when patients are given a regular dose, it is to keep things simple. For example, paracetamol (acetaminophen) is usually given in doses of 500mg per pill. The standard dose is two pills to be taken every four to six hours, with a maximum consumption of eight pills per day. Higher levels may give more pain relief, but the rate at which it is metabolized gives a risk of toxicity if these limits are exceed – the spacing between doses, and daily limit, avoid excess exposure.

To address the challenge of effectively managing patients' medication usage, one option is to produce personalized medication plans. Personalized medicine is defined as providing "the right patient with the right drug at the right dose at the right time" (Sadee and Dai 2005). Historically, the scope for this has been limited to where it is essential (for instance, personalized insulin regimes for diabetics) but is recently becoming more viable through the uptake of technology – at one extreme, with the use of a drug dosage printer to 'print' drugs with specific doses (Hirshfield et al. 2014).

In this paper we detail the application of PDDL+ to planning a patient's daily routine, in terms of the consumption of pain relief medication, and activities for the day; including considering how different activities require different levels of pain relief, for the patient's comfort. The metabolism of medication is non-linear (negative exponential), and many otherwise-effective PDDL planners do not support non-linear domains. To address the limited performance of PDDL+ planners on the non-linear model, we detail a linearize–validate cycle, where a solution to an initial linear approximation is found using the planner; then validated against the non-linear version of the domain using the plan validator, VAL (Howey, Long, and Fox 2004). If the solution does not validate, we use the output from VAL to refine the approximation, and plan again; repeating this process until a solution plan that is valid when considering the full non-linear pharmacokinetics is found. We consider two ways to refine the linearization, with a trade-off between linearized model accuracy in theory, and planner performance in practice. We evaluate our approach on problems from the application, developed through working with a physician.

2 Planning Background

The logical basis for temporal planning, as modeled in PDDL2.1 (Fox and Long 2003), is a collection of propositions P, and a vector of numeric variables $\mathbf{v}$. These are manipulated and referred to by actions. Actions can be applied if their preconditions are satisfied. A single *condition* is either a single proposition $p \in P$, $\neg p$, or a numeric constraint over $\mathbf{v}$. For our domain, it suffices to assume all such constraints are linear, and hence can be represented in the form:

$$\mathbf{w}.\mathbf{v}\{>, \geq, <, \leq, =\}c$$

($\mathbf{w}$ is a vector of constants and c is a constant). We assume a *precondition* is a conjunction of zero or more conditions.

Each durative action A has three sets of preconditions: $\text{pre}_{\vdash} A$, $\text{pre}_{\leftrightarrow} A$, $\text{pre}_{\dashv} A$. These represent the conditions that must hold at its start, throughout its execution (invariants), and at the end, respectively. Instantaneous effects can occur at the start or end of A: $\text{eff}^{+}_{\vdash} A$ ($\text{eff}^{-}_{\vdash} A$) denote propositions added (resp. deleted) at the start; $\text{eff}^{num}_{\vdash} A$ denotes any nu-

meric effects. Similarly, $\text{eff}^{+}_{\dashv} A$, $\text{eff}^{-}_{\dashv}$ and $\text{eff}^{num}_{\dashv}$ record effects at the end. We assume all such effects are of the form:

$$v\{\texttt{+=}, \texttt{-=}, \texttt{=}\}\mathbf{w}.\mathbf{v} + c \qquad \text{where } v \in \mathbf{v}$$

Semantically, the values of these instantaneous effects become available small amount of time, ϵ, after they occur. Durative actions additionally have a duration constraint: a conjunction of numeric constraints applied to a special variable `?duration`, denoting its duration.

PDDL+ augments PDDL problems with processes and events which, too, have preconditions, and effects. Events are discrete, like instantaneous actions: if an event's preconditions are satisfied, it occurs; and the event's effects happen. Processes are closer to durative actions, with $\text{pre}_{\leftrightarrow} A$ corresponding to the process' precondition; then, while $\text{pre}_{\leftrightarrow} A$ is satisfied, its effects occur. These are continuous in nature, each of the form:

$$dv/dt\texttt{=}\mathbf{w}.\mathbf{v} + c$$

If all the weights in $\mathbf{w}$ are 0, then the effect is linear (constant gradient) with rate of change c. Otherwise, the effect is non-linear, varying according to the specified first-derivative.

The critical distinction between processes and events, and actions, is that a process/event *must* occur as soon as its precondition is satisfied, whereas actions *may* occur when their preconditions are satisfied. The task of planning is thus to find time-stamped actions, with associated durations, such that all actions' preconditions are satisfied; and when executed, accounting for any processes and events that may be triggered, a state is reached in which the goals are satisfied, and no actions are executing.

PDDL+ planners can be characterized by the form of continuous change that they support. OPTIC (Benton, Coles, and Coles 2012; Coles and Coles 2014) relies on a Mixed Integer Programming (MIP) solver to reason with the interaction between time and numeric variables, so can handle only linear problems. SMTPLAN (Cashmore et al. 2016) encodes the planning task in SAT Modulo Theories, using a theory that can handle polynomial continuous numeric change – this allows it to support a subset of non-linear domains but not, for instance, negative exponential continuous numeric change. UPMURPHI (Penna et al. 2009), Dino (Piotrowski et al. 2015) and ENHSP (Scala et al. 2016) have no restrictions on the form that continuous numeric change can take. UPMURPHI (Penna et al. 2009) and Dino (Piotrowski et al. 2015) use a discretize–validate cycle, where time is discretized; the problem is solved using a model checker; and if the solution returned is not valid (according to VAL (Howey, Long, and Fox 2004)), the discretization is refined, and the planning cycle continues. This provides motivation for the concept of iterative refinement that we will use as the basis for the approach we set out in this paper.

3 Application Description

Our target application is a patient-support system to help plan their daily routine, considering their daily goals, and managing their medication levels; specifically pain relief. These two are inter-dependent: different activities require different levels of pain relief, to ensure the patient's comfort. In this section we will describe the mechanics of our application domain, and how a model of it is built in PDDL+.

3.1 Pharmacokinetics

When consumed, medication is metabolized in the body over time, leading to a decay of the active medication level. While pharmacokinetics are complex, a reasonable model is to assume negative-exponential (i.e. first-order) decay, with drug-dependent half-lives depending on the rate at which the active ingredients are metabolized (Geenen et al. 2013). In this paper we will focus on pain relief management with a single painkiller as an exemplar problem, so can discuss drug levels in terms of desired levels of pain relief (pr). Returning to the example of paracetamol, the half life is up to 3 hours. That means if someone takes 1000mg of paracetamol at 12.00pm, there will be 500mg of the drug left in three hours (i.e. at 3.00pm). In another three hours (i.e. 6.00pm), there will be 250mg of the drug left, and so on.

Since these pharmacokinetics are non-linear, our decay processes is modeled as a PDDL process with negative exponential change upon the level of pain relief the patient experiences. Below is the decay process, where k represents the decay constant and pr is the pain-relief variable that is continuously changing:

```
(:process decay
 :parameters ()
 :precondition
   (> (pr) 0)
 :effect
   (decrease (pr) (* #t (* (pr) (k))))
)
```

The precondition in the above process means that when pr is greater than zero, the process is active. For a given medication, with half-life $t_{1/2}$, the decay constant k is:

$$k = \frac{\ln 2}{t_{1/2}}$$

3.2 Consuming Medication

There is one action that changes pr: `consume`. This is a durative action with duration gap and the following preconditions and effects:

- To start the action, a proposition *safe-to-consume* (true initially) must be true; and a variable *doses* (0 initially) must be less than m (the maximum number of doses).
- When started, *safe-to-consume* is deleted; *doses* is increased by 1; and the pain relief level pr is increased by the dose of medication.
- At the end of the action, *safe-to-consume* is added

```
(:durative-action consume
  :parameters ()
  :duration (= ?duration (gap))
  :condition (and
    (at start (safe-to-consume))
    (over all (can-do-normal-actions))
```

```
  )
  :effect (and
    (at start (increase (pr) (dose)))
    (at start (increase (doses) 1))
    (at start (not(safe-to-consume)))
    (at end (safe-to-consume))
  )
)
```

Effectively, *safe-to-consume* and *doses* perform the requisite book-keeping to enforce the constraints on maximum medication consumption. *safe-to-consume* acts as a semaphore: no two consume actions can overlap; and the duration gap serves to ensure the minimum specified time between doses is respected. *doses* is a simple counter, with a capped limit in the problem goal, to ensure that if doses are repeatedly taken, the maximum safe limit for the period over which we are planning cannot be exceeded.

We note that it is somewhat of a simplification to assume the increase in pain relief occurs instantly at the start of the action. For our application, this is sufficient as the margin of error when humans execute their plans would be greater than the minor difference to timings that would occur if the model was refined.

3.3 Activities

Alongside the consume action, we have several actions that represent various activities that a patient would like to do throughout the day; all of which would have some dependence on the amount of pain relief the patient experiences. These actions run independently of the consume action.

Activities include:

- `eat` – the act of consuming a meal. The goal of the problem sets how many times this needs to be achieved, according to the patients needs.
- `drive` – driving between locations at which activities need to be performed.
- `exercise` – that the patient gets some exercise. For our purpose, we assume they need to be at a gym with suitable facilities for their needs. The goal of the problem sets how many times this needs to be achieved.

Each of these actions have a precondition set that requires a minimum amount of pain-relief (or pr value). For example, the `exercise` actions is modeled as follows, with the fact *free* preventing two activities from occurring at the same time:

```
(:durative-action exercise
  :parameters (
    ?p - patient
    ?g - gym
  )
  :duration (= ?duration 90)
  :condition (and
    (over all (is-at ?p ?g))
    (over all (>= (pr) 300))
    (at start (free))
  )
  :effect (and
    (at start (not (free)))
    (at end (free))
    (at end (went-to-gym))
  )
)
```

3.4 Global constraint on pain relief

As a global constraint, we require that the patient has a minimum level of pain relief at all times – regardless of what actions are executing. To capture this in PDDL+, we use an event that deletes an auxilary fact if the pain relief threshold falls below a prescribed minimum. As this fact cannot be added by any other event or action in the domain, and is required as a goal, it means any plan in which the event fires is invalid; and hence the planner must respect the minimum level of pain relief.

The event is written as follows:

```
(:event prfailure
  :parameters ()
  :precondition (and
    (<= (pr) (min-pr))
    (pr-check))
  :effect (not (pr-check))
)
```

3.5 Planning for a given period

As described thus far, a valid plan would be one that performs sufficient activities to satisfy the goals, then finishes. As our application requires us to plan a patients activities and medication for a given period, for instance a day, this is not sufficient.

To insist that the plan must satisfy the global pain relief constraint over a desired horizon finishing at time t, we use a Timed Initial Literal (Hoffmann and Edelkamp 2005) (TIL) that deletes *can-do-normal-actions* adds an auxiliary fact at time t; with this fact then being a goal[1]. While the goal is obviously trivial, this has the indirect effect of ensuring sufficient consume actions are planned before time t, to avoid the pain relief falling below the minimum acceptable level, and hence the minimum-check event firing.

4 Initial Evaluation

Having defined the domain for our application, we considered which available planners could be used for it. Given it has a negative-exponential process (decay) a limited range were available: UPMurphi (Penna et al. 2009), DiNo (Piotrowski et al. 2015) and ENHSP (Scala et al. 2016). As ENHSP does not support durative actions, an equivalent formulation had to be produced, where each durative action was compiled into an instantaneous action to denote its start; a process to mark the time since it started; and an event to denote its end. For instance, consume became:

[1] Alternatively, with PDDL+ one could encode the TIL by using a process that measures time elapsed, and an event that fires at the appropriate time. We use a TIL model as it is a more intuitive encoding, though we needed to use the PDDL+ alternative with ENHSP in our evaluation.

```
(:action consume-start
  :parameters ()
  :precondition (and
    (safe-to-consume)
    (can-do-normal-actions)
    (not (consume-running))
  )
  :effect (and
    (increase (pr) (dose))
    (increase (doses) 1)
    (not (safe-to-consume))
    (increase (executing-actions) 1)
    (consume-running)
    (assign (consume-tick) 0)
  )
)

(:process consume-running-ticker
 :parameters ()
 :precondition (and
   (consume-running)
   (can-do-normal-actions))
 :effect (increase (consume-tick)
                   (* #t 1))
)

(:event consume-end
 :parameters ()
 :precondition (and
   (consume-running)
   (= (consume-tick) (gap)))
 :effect (and
   (safe-to-consume)
   (not (consume-running))
   (decrease (executing-actions) 1)
 )
)
```

Similarly, for timed initial literals, a process was used to count the time that had elapsed; and an event created set to fire at the appropriate time, whose effects were set to those of the corresponding TIL.

Regardless of whether this reformulation was used, or the PDDL model with durative actions described in Section 3, neither UPMurphi, Dino, nor ENHSP were able to solve any problem whose solution needed medication consumption and activity actions. UPMurphi and Dino both returned solutions, but the solutions were always invalid according to the plan validator, VAL (Howey, Long, and Fox 2004). ENHSP reported the problems to be unsolvable, even though valid hand-written solutions were known to exist.

Scaling back the domain to contain just the consume actions – i.e. no activities were needed – gave problems that were still not (validly) solvable by UPMurphi or Dino, but were solvable by ENHSP. A problem where the goal was to maintain a minimum pain relief level over a 21-hour period, necessitating 4 consume actions, could be solved in 14.33 seconds. This suggests that ENHSP is in principle able to handle the numeric dynamics of the domain, but has difficulty scaling to plans that need more actions, with causal dependencies. For consume, its preconditions are always satisfied once sufficient time has passed, so there is no need to apply other actions in order to support its application; whereas if activities are included, where there are causal planning decisions to be made to find a solution plan, it can no longer find a solution.

5 Planning using a Linearize → Validate Cycle

Due to the limited performance observed when using planners capable of reasoning natively with the negative exponential decay process, we considered how we might be able to use other PDDL+ planners – namely OPTIC and SMTPlan – that can reason with a linear model. As the pharmacokinetics are inherently non-linear, we devised a linearize–validate cycle, the intuition being that if we iteratively refine a linear approximation until a solution is found that validates – according to the non-linear model – these other planners can be used.

As a high-level overview, before going into the details, our cycle proceeds as follows:

- An initial linearization of the PDDL+ model is made, fitting a single line to each process, spanning between the (specified) lower and upper bounds of the variables it affects
- This linear model is solved by a planner; and the solution found is validated by VAL
- If the resulting solution is valid, we return the solution.
- Otherwise, the diagnostic trace from VAL guides the refinement of the linearization – the incumbent process linearizations are split into multiple pieces, whose boundaries are based on the values of variables as seen in the diagnostic trace. The intuition here is that if the plan was invalid, it must either have had an unsatisfied precondition, or an unanticipated event or process occurred. If any of these happens, the diagnostic trace will report the (calculated) values of variables at that time; so by using these to refine the linearization, we aim for it to be accurate around the values that appear to be relevant to plan construction.
- The cycle continues, with the refined linear model passed to the planner.

Assuming there is a solution plan to be found, then in theory, the cycle is guaranteed to find one: the linearization is a relaxation, as it over-estimates pain relief. After a number of iterations of finding successive plans that were invalid according to the non-linear domain, leading to refinement of the linearization, it will become closer and closer to the non-linear domain. The planner used might struggle to scale to a model with so many processes, but in theory at least, it is complete.

5.1 Initial Linearization

As a starting point, we must devise an initial linear approximation. For this, we refer to the given upper and lower

bounds – ub_v and lb_v on each variable v. Note these bounds must be sensitive to the underlying mathematics: as the decay process is negative-exponential, the lower bound cannot be 0, as the time taken to go from ub_v to 0 would be infinite. For pain relief, the upper bound equates to the patient taking all their medication at once. For each process, we calculate $\overline{r_{ub_v,lb_v}}$, the average gradient on the change on v from ub_v to lb_v as:

$$\overline{r_{ub_v,lb_v}} = \frac{ub_v - lb_v}{TGF_{(ub_v,lb_v)}}$$

...where $TGF_{(ub_v,lb_v)}$ is the time to go from ub_v to lb_v. As the change by the decay process is a negative exponential, $TGF_{(ub_v,lb_v)}$ is:

$$TGF_{(ub_v,lb_v)} = \frac{-t_{1/2}}{\ln 2} \cdot \ln(\frac{lb_v}{ub_v})$$

...where $t_{1/2}$ is the half life. In other cases, which are outside the scope of this work, the figures could be derived using the numeric analysis techniques provided as part of VAL.

The initial approximation for the decay process is depicted in Figure 1. In the context of our application, the solid line shows the negative-exponential change in pr, assuming pr at time 0 is ub. The x-axis ranges from 0 until the time at which pr would reach lb. The dotted line shows a linear approximation spanning this time, with gradient $\overline{r_{ub,lb}}$.

A substantial caveat is that the initial linear approximation substantially over-estimates pr. If there is a minimum pr threshold (e.g. the dashed line in Figure 1), then the actual value of pr will fall below the threshold far sooner than would be considered to be the case according to the linear approximation. But, we have at least turned a negative exponential into a simple linear process, with a constant (linear) effect, that can be given to a suitable planner:

```
(:process decay_ub_lb
 :parameters ()
 :precondition
   (and (>= (pr) lb) (<= (pr) ub))
 :effect
   (decrease (pr) (* #t r_ub_lb))
)
```

5.2 Refining the Linearization

Solving this linearized problem, then validating the plan against the non-linear model using VAL (Howey, Long, and Fox 2004), will identify where the inaccuracies inherent in the linearization have caused issues. This arises in one of two ways:

- A precondition on a continuously changing variable, or a goal, was unsatisfied, and hence the solution plan is invalid. This would occur, for instance, if an activity needed a minimum pain relief level, but this was not actually available.
- An event or process occurred that did not occur when using the linear model; or vice versa. If we are lucky, the solution plan is still valid regardless. But, the events and processes may change the propositions and the values of variables in the solution state trajectory, making the solution plan invalid.

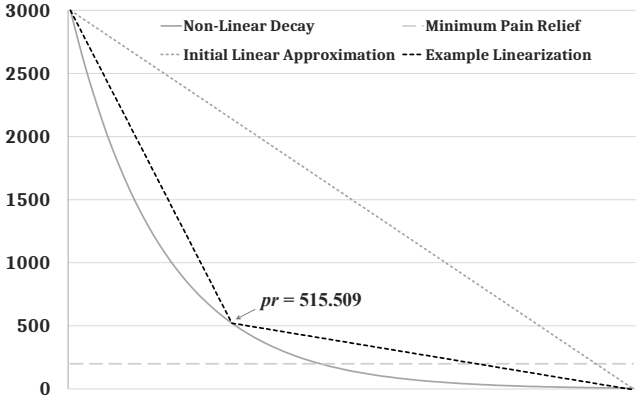

Figure 1: Example Pain Relief Process & Linearizations

In both of these cases, VAL produces a diagnostic trace: a time-stamped progression through the plan, including what the value of each variable was at each happening in the plan, as evaluated against the non-linear domain. With this information, we can refine the linearization: for a variable v, instead of having a single-segment linear process spanning the whole range ub_v to lb_v, we can have several processes each covering one segment of this range.

Our motivation for refining the linearization to give the right value of v at happenings is based on the observation that the error in the linearization is acceptable, as long as it gives the right value when it matters; i.e. when a precondition referring to v is checked, or when processes/events with preconditions to v should occur/not occur. Hence, we ensure that on each iteration, the linearization is refined based on the observed values of variables. This does not guarantee that a solution to the non-linear model will be found on the second iteration, but it does mean the model is iteratively refined to exclude apparently attractive but actually infeasible solutions.

We propose two approaches, the first of which is shown in Algorithm 1. We begin by running the planner to find a solution based on the initial approximation; i.e. starting with the initial values of ub_i and lb_i for each variable v_i. Hence, at line 7, when the linearized model P' is generated, we have only one pair of bounds in the set for each variable, and for each non-linear process generate a single linear process covering this range, with an effect with gradient $\overline{r_{ub,lb}}$. A solution Π to P' is then found.

As noted earlier, it is likely that when using the initial linearization, Π will not be a solution to P. Hence, to refine the linearization, we refer to the happenings in the diagnostic trace from VAL, an example of which can be seen in Figure 2, and keep all calculated values of variables (in this case, pr, marked in boldface); i.e. for the plan Π, what values of pr were seen according to the model P. Each of these is added to the set of bounds (line 11) for the variable con-

Algorithm 1: Linearize–Validate Cycle

Data: P, the non-linear planning domain and problem;
$bounds = [\{ub_0, lb_0\}, .., \{ub_n, lb_n\}]$, the initial bounds on variables $v_0..v_n$
Result: A solution plan, Π

1 $P_{base} \leftarrow P$, with non-linear processes removed;
2 **while** *true* **do**
3 $P' \leftarrow P_{base}$;
4 **foreach** $v_i \in [v_0..v_n]$ **do**
5 $sorted \leftarrow$ [$bounds_i$, sorted in descending order];
6 **foreach** *process affecting* v_i **do**
7 add a linear process on v_i to P' for each successive pair in $sorted$;
8 $\Pi \leftarrow$ solve P' using planner;
9 **if** Π *is a valid solution to* P **then return** Π ;
10 **foreach** $v_i = c \in$ *VAL's diagnostic trace for* Π *using model* P **do**
11 $bounds_i \leftarrow bounds_i \cup \{c\}$

```
Checking next happening (time 240)
Updating (pr) (1300) by 515.905 assignment

Checking next happening (time 240)
Adding (safe-to-consume)

Checking next happening (time 486.079)
Updating (pr) (515.905) by 200 assignment

EVENT triggered at (time 486.079)
Triggered event (prfailure)
Deleting (pr-check)

Checking next happening (time 544.824)
Updating (pr) (200) by 159.509 assignment

...

Checking next happening (time 784.824)
Updating (pr) (1159.51) by 460.152 assignment
```

Figure 2: Example Diagnostic Trace from VAL

cerned. With these updated sets of bounds, the loop starts again, generating an updated linear problem P', and once again attempting to find a solution plan. For the updated problem, the bounds are sorted, and processes generated for each adjacent pair of bounds, each with its own $\overline{r_{ub,lb}}$ value. This collection of processes uses a "stacked" model to generate a combined linear effect (Denenberg and Coles 2018; 2019).

5.3 Cherry Picking Bounds for the Linearization

When we initially look at the diagnostic trace from VAL, we will initially find several values for each variable, and following Algorithm 1 create an extensive well-refined linearization with several segments for each process. However, the *more* refined an approximation, the *longer* the planner takes to find a solution.

We will demonstrate this with an example. Let us assume that we have a domain with a value that experiences non-linear continuous change. Our upper bound is 1000 and our lower bound is 0. As described in Section 5.1, we devise an initial linearization; solve it; validate the solution; then find values from VAL's diagnostic trace to create an extensive linearization. If we found the values 460.512, 200 and 515.509 in the diagnostic trace, the refinement will consists of segments that reflect the linear change between each adjacent pair of of values (ub, 515.509, 460.512, 200, lb). Switching between these processes carries overheads for the planner; and even if the additional accuracy gained by adding all the bounds was beyond what was needed to find a solution, these overheads are still incurred.

Inspired by this, we present a second approach – Algorithm 2 – that 'cherry picks' the values from VAL's diagnostic trace, aiming to add just a small subset of them to the linearization, such that the solution obtained on the last iteration is not a valid solution to this refined linearization. The key detail is that at line 13, it loops through the power set of all additional bounds values that were seen in the last diagnostic trace. Importantly, these are sorted into ascending size order, according to set cardinality, thereby trying to find a small set of additional bounds that is sufficient to reject the previous candidate solution plan. As soon as such a set of bounds is found, these are kept as the bounds for the next iteration (line 23) and the outer loop starts again, calling the planner using a linearization based on these updated bounds. As an example, Figure 1 shows a linearization derived from a cherry-picked bound of 515.509, added to the bounds set alongside the initial upper and lower bound.

6 Evaluation

Having defined our linearize–validate cycle, we are now able to evaluate its efficacy using as a kernel PDDL+ planners capable of reasoning with linear processes: OPTIC (Coles, Coles, and Benton 2012) and SMTPlan (Cashmore et al. 2016). Of these, only OPTIC could solve any problems – SMTPlan could not – so we will use this in our evaluation.

6.1 Consume-only problems

We begin with consume-only problems, i.e. there are no activities to be planned, and the task is to maintain a minimum pain relief level across the day, over some specified horizon (denoted by a Timed Initial Literal). These have less propositional causal reasoning, but can be solved by ENHSP (as noted in Section 4).

We first evaluate the scalability of the linearize–validate cycle when using the extensive linearization, i.e. no cherry picking, described in Section 5.3. The results of this can be seen in Table 1. The planning horizons tested started at 540 minutes (i.e. 9 hours), and increased by 60 minutes each time; extending the horizon necessitates consuming more doses of medication. 'total' gives the total planning time; and 'tinitial' and 'trefined' the planning time for the first and

Algorithm 2: Linearize–Validate Cycle with Cherry Picking

Data: P, the non-linear planning domain and problem;
$bounds = [\{ub_0, lb_0\}, .., \{ub_n, lb_n\}]$,
$sublists = [[]]$, the initial bounds on variables $v_0..v_n$
Result: A solution plan, Π

$P_{base} \leftarrow P$, with non-linear processes removed;
while *true* **do**
  $P' \leftarrow P_{base}$;
  foreach $v_i \in [v_0..v_n]$ **do**
    $sorted \leftarrow$ [$bounds_i$, sorted in descending order];
    foreach *process affecting* v_i **do**
      add a linear process on v_i to P' for each successive pair in $sorted$;
  $\Pi \leftarrow$ solve P' using planner;
  if Π *is a valid solution to* P **then return** Π ;
  $newbounds \leftarrow \emptyset$;
  foreach $v_i = c \in$ *VAL's diagnostic trace for* Π *using model* P **do**
    if $c \notin bounds_i$ **then**
      $newbounds \leftarrow newbounds \cup \{\langle v_i, c\rangle\}$;
  foreach $nb \in \mathcal{P}(newbounds)$ *in increasing size order* **do**
    $bounds' \leftarrow bounds$;
    foreach $\langle v_i, c\rangle \in nb$ **do**
      ${bounds'}_i \leftarrow {bounds'}_i \cup \{c\}$;
    $P' \leftarrow P_{base}$;
    foreach $v_i \in [v_0..v_n]$ **do**
      $sorted' \leftarrow$ [$bounds'_i$, sorted in descending order];
      foreach *process affecting* v_i **do**
        add a linear process on v_i to P' for each successive pair in $sorted'$;
    if Π *is not a valid solution to* P' **then**
      $bounds \leftarrow bounds'$;
      break;

second iteration respectively. Where the solution found by the initial approximation was incidentally a valid solution to the non-linear model, 'trefined' figures are not given. No more than two iterations were needed on any problem.

As a headline observation, the planning time here is strongly correlated with the number of doses to be taken – this is shown more clearly in Figure 3, where the planning time (left Y-axis) tracks the number of doses needed (right Y-axis). This happens due to the planner needing to switch the linear process segments on when medication is consumed; and then off, one by one, as the pain relief level falls; until the next dose is taken and they are switched on again. Although not part of the explicit solution returned by the planner, these process switches are part of the planner's

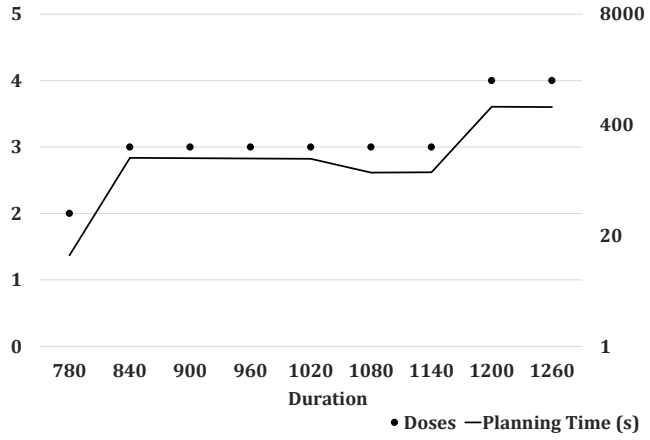

Figure 3: Comparison Between Doses Needed and Planning Time (in seconds), when using the Extensive Linearization

search space, increasing the size of the search space. On the harder problems in particular, the time taken is much longer than would be reasonable in practice.

We next evaluate our 'Cherry Picking' algorithm, described in Section 5.3, again using consume-only problems. The results of this, compared to the Extensive refinement (keeping all bounds) can be seen in Table 2. As can be seen, cherry picking bounds leads to a dramatic reduction in planning time compared with the Extensive refinement. This is due to the Cherry Picked refinement having fewer bounds – as can be seen on the largest problem, even a modest reduction in the size of the bounds set (from 5 to 4) leads to a dramatic reduction in overall planning time. As an aside, we note this was also able to out-perform starting with a hand-coded 'best guess' initial set of bounds, as the right bounds needed for a good linearization varies from problem-to-problem – the Cherry Picking approach provides an efficient way of determining where these should be.

As ENHSP is able to solve consume-only problems, as a final test here we can compare OPTIC with the cherry-picking wrapper, with ENHSP. The results for this can be seen in Table 3. The time taken to find plans shows that our approach is competitive with ENHSP on consume-only problems, with neither planner dominating the other.

6.2 Problems with Activities

Having shown our linearize–validate cycle, with OPTIC, is competitive on consume-only problems, we turn our attention to more interesting problems that require both activity planning (as per Section 3.3), and medication consumption planning – i.e. the problems that could not be solved at all with existing planners in Section 4. We use only the 'Cherry Picking' algorithm in these tests, as using the extensive linearization (keeping all bounds) could not solve a single problem.

As there is nothing to which we can compare our performance, we perform a sensitivity analysis on our performance. As a baseline configuration:

- The patient needs to have at least one meal;

	Planning Horizon (seconds)												
	540	*600*	*660*	*720*	*780*	*840*	*900*	*960*	*1020*	*1080*	*1140*	*1200*	*1260*
$\mathbf{t_{initial}}$	0.05	0.03	0.03	0.04	0.03	0.04	0.03	0.03	0.06	0.06	0.06	0.09	0.09
$\mathbf{t_{refined}}$	0.71	-	-	-	11.72	164.19	162.99	161.29	160.09	110.05	111.2	654.77	649.86
total	0.76	0.03	0.03	0.04	11.75	164.23	163.04	161.32	160.15	110.11	111.26	654.83	649.95
doses	2	2	2	2	2	3	3	3	3	3	3	4	4

Table 1: Planning times (seconds) when using the Extensive Linearization, increasing Plan Horizon

Horizon		**Extensive**	**Cherrypick**
540	Total Bounds	*2*	*2*
	Planning Time	*0.76*s	*0.76*s
900	Total Bounds	*6*	***3***
	Planning Time	*163.04*s	***0.21***s
1260	Total Bounds	*5*	***4***
	Planning Time	*649.95*s	***16.48***s

Table 2: Performance of the Extensive and Cherry Picked Linearizations on Consume-Only Problems

Horizon		**OPTIC**	**ENHSP**
540	Planning Time (s)	*0.76*	*5.42*
900	Planning Time (s)	*0.21*	*9.25*
1260	Planning Time (s)	*16.48*	*14.33*

Table 3: Performance Comparison on Consume-Only Problems: Cherry-Picked Linearized Domain with OPTIC vs Non-linear Domain with ENHSP

- The patient needs to exercise;
- There are only two locations on the map.

Varying these parameters within ranges requested by a physician, we considered planning for one or two meals; and considering the number of locations a patient might need to visit in the day, we increased the number of locations from two to four.

The results of this scalability analysis can be seen in Table 4. As a summary observations: we can solve within no more than 30 seconds problems that were unsolvable by prior PDDL+ planners; over a range of parameter values that are flexible enough to meet the needs of patients.

To make some more scientific observations, first, comparing the performance in the Baseline problem, with the consume-only performance from Table 3, it can be seen that adding activities does not dramatically affect the time taken to solve the problems. Looking at OPTIC's explored search space, its heuristic was effective for the propositional causal reasoning needed to plan these activities (as one would see in PDDL2.1); but was relatively less effective for resolving the PDDL+ constraints. Hence, adding activities had minimal impact on planning time, compared with consume-only problems.

Observing the effect of changing the number of meals, planning for two meals is slightly quicker than planning for one meal. This is due to a quirk of search. Because 'eat' actions have preconditions on pain relief, this strengthens

Horizon	**Results**	**# Meals**		**# Map locations**			**Baseline**
		1	2	2	3	4	
600	# VAL Calls	3	4	3	3	3	3
	Planning Time	9.48	8.38	9.48	10.02	10.60	9.48
840	# VAL Calls	3	4	3	3	3	3
	Planning Time	16.07	8.25	16.07	16.89	18.96	16.07
1080	# VAL Calls	3	4	3	3	3	3
	Planning Time	16.36	12.57	16.36	20.26	26.35	16.36

Table 4: Sensitivity analysis of planning time, varying problem parameters and planning horizon. Baseline has 1 meal and 1 gym location. Planning times are in seconds.

the planner's causal reasoning that medication consumption is needed. In the two-meal case, this is sufficient to allow the heuristic to include a consume action in the relaxed plan. For a single meal, this is not the case, so the heuristic is less informative, increasing planning time.

Varying the number of map locations varies planning time, due to the increased planning branching factor. The effect of this increases as the planning horizon goes up, as there is a longer period of time over which redundant drive actions can be planned. Finally, we note that Table 4 shows the number of VAL calls made as part of the cherry-picking process. In practice, very few calls were needed to identify an additional bound that made the previous solution invalid – all problems were solved with just two iterations, and a handful of VAL calls that allowed just a single bound to be cherry-picked. Although not observed in our application problems, in theory, if a pathological case was generated – generating a large set of candidate bounds, so that looping over the power set was prohibitively expensive – a fall-back option would be to keep all bounds and use a linear-time algorithm to consider their removal one-by-one, keeping a bound if its removal makes the plan found using the previous linearization be a valid solution to the new linearization. In any case, obtaining a large number of candidate bounds would require a large number of happenings, well beyond the scale of prior state-of-the-art work in the area.

7 Discussion and Future Work

As discussed in the Evaluation, our linearize–validate cycle, with OPTIC as its kernel, is able to solve realistic problems generated for our application, following the advice of the physician with whom we are working.

Having made a contribution to the case of managing a single medication, for future work we will progress to look at polypharmacy. As almost a quarter of the UK population are on at least three prescriptions (Scholes, Faulding, and Mindell 2014), this is a substantial area of interest.

In the case of painkillers, these are often complementary. If a patient was only on one drug, it may be difficult to give adequate pain relief due to the constraints of the drug itself. For example, paracetamol is an effective painkiller, but the dosage restrictions mean it cannot always be used as a monotherapy to give sufficient pain relief. Thus, using multiple painkillers (for example paracetamol and ibuprofen) gives greater potential for pain management (taking both at the same time), and greater flexibility (taking them at different times). The modifications to the model to support this are relatively straightforward: rather than using a single pr variable, use multiple such variables (one for each painkiller) and define conditions on pain relief to refer to a weighted sum of these.

A more challenging case is where there are interactions between medication. Ideally, two adversely interacting medications would not be taken concomitantly, but it is sometimes unavoidable. To handle pharmacokinetic interactions (one drug affects the rate of metabolism of another), the drug decay processes would need to be updated. How to do this well remains an open challenge.

8 Conclusions

In this paper, we described an application of planning to a patient's medication needs and daily activities. Modeled in PDDL+, the problems for this application were unsolvable by previous state-of-the-art planners. Hence, we devised a linearize–validate approach for solving these problems, iteratively refining a linear approximation of the domain using the diagnostic trace returned by the plan validator, VAL.

Our wrapper, when used as a wrapper around the planner OPTIC, was shown to be an effective solution to the problems in this domain. This is an exciting result, opening the range of applications amenable to PDDL+ planning, and hence for planning to make a contribution to this important area.

References

Benton, J.; Coles, A. J.; and Coles, A. I. 2012. Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the Twenty Second International Conference on Automated Planning and Scheduling (ICAPS)*.

Cashmore, M.; Fox, M.; Long, D.; and Magazzeni, D. 2016. A Compilation of the Full PDDL+ Language into SMT. In *Proceedings of the Twenty-Sixth International Conference on International Conference on Automated Planning and Scheduling (ICAPS)*.

Coles, A. J., and Coles, A. I. 2014. PDDL+ Planning with Events and Linear Processes. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS)*.

Coles, A. J.; Coles, A. I.; and Benton, J. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling*.

Denenberg, E., and Coles, A. 2018. Modelling Sequences of Processes in PDDL+ for Efficient Problem Solving. In *Proceedings of the Workshop on Knowledge Engineering for Planning and Scheduling (KEPS) at the 28th International Conference on Automated Planning and Scheduling (ICAPS)*.

Denenberg, E., and Coles, A. 2019. Mixed Discrete Continuous Non-Linear Planning Through Piecewise Linear Approximation. In *Proceedings of the Twenty-Ninth International Conference on International Conference on Automated Planning and Scheduling (ICAPS)*.

Fox, M., and Long, D. 2003. PDDL2.1 : An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*.

Geenen, S.; Yates, J. W. T.; Kenna, J. G.; Bois, F. Y.; Wilson, I. D.; and Westerhoff, H. V. 2013. Multiscale modelling approach combining a kinetic model of glutathione metabolism with pbpk models of paracetamol and the potential glutathione-depletion biomarkers ophthalmic acid and 5-oxoproline in humans and rats. *Integrative Biology* 5:877–888.

Hirshfield, L.; Giridhar, A.; Taylor, L.; Harris, M.; and Reklaitis, G. 2014. Dropwise additive manufacturing of pharmaceutical products for solvent-based dosage forms. *Journal of Pharmaceutical Sciences* 103:496–506.

Hoffmann, J., and Edelkamp, S. 2005. The Deterministic Part of IPC-4: An Overview. *Journal of Artificial Intelligence Research*.

Howey, R.; Long, D.; and Fox, M. 2004. VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning using PDDL. In *The 16th IEEE International Conference on Tools with Artificial Intelligence*, 294–301.

Penna, G. D.; Magazzeni, D.; Mercorio, F.; and Intrigila, B. 2009. UPMurphi: A Tool for Universal Planning on PDDL+ Problems. In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS)*.

Piotrowski, W.; Fox, M.; Long, D.; Magazzeni, D.; and Mercorio, F. 2015. Heuristic Planning for Hybrid Systems. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*.

Sadee, W., and Dai, Z. 2005. Pharmacogenetics/genomics and personalized medicine. *Human Molecular Genetics*.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramirez, M. 2016. Interval-based relaxation for general numeric planning. In *Proceedings of the European Conference on Artificial Intelligence*, 655–663.

Scholes, S.; Faulding, S.; and Mindell, J. 2014. Use of prescribed medicines. In *Health Survey for England – 2013*. NHS Digital.

The Academy of Medical Sciences. 2014. Patient Adherence to Medicines.